420-KEF – Travail pratique #1 – 25% de la note finale – par équipe de deux – remise le 11 avril

Compétences

- Observer/modifier les composants d'un projet ASP.Net MVC
- Comprendre l'architecture MVC
- Exploitation d'une base de données Ms SQL
- Utilisation d'entity Framework en tant que DAL
- Utilisation de Linq pour gérer les résultats de requête à la BD
- Créer/Modifier des modèles
- Créer/Modifier des contrôleurs
- Exploitation des objets serveur (HttpContext, Session, etc.)
- Ajouter/modifier des actions à un contrôleur
- Produire et modifier des formulaires Web CRUD de vue
- Assurer la validation des données saisies dans un formulaire Web
- Comprendre et modifier la page maîtresse des pages Web

Énoncé

Vous devez bâtir un site transactionnel de gestion de favoris (bookmark) avec authentification.

Démo du projet à construire : http://www.informatique.clg.qc.ca/Bookmarks/

Pages à construire

Usager anonyme :

- Créer un compte
- Connection

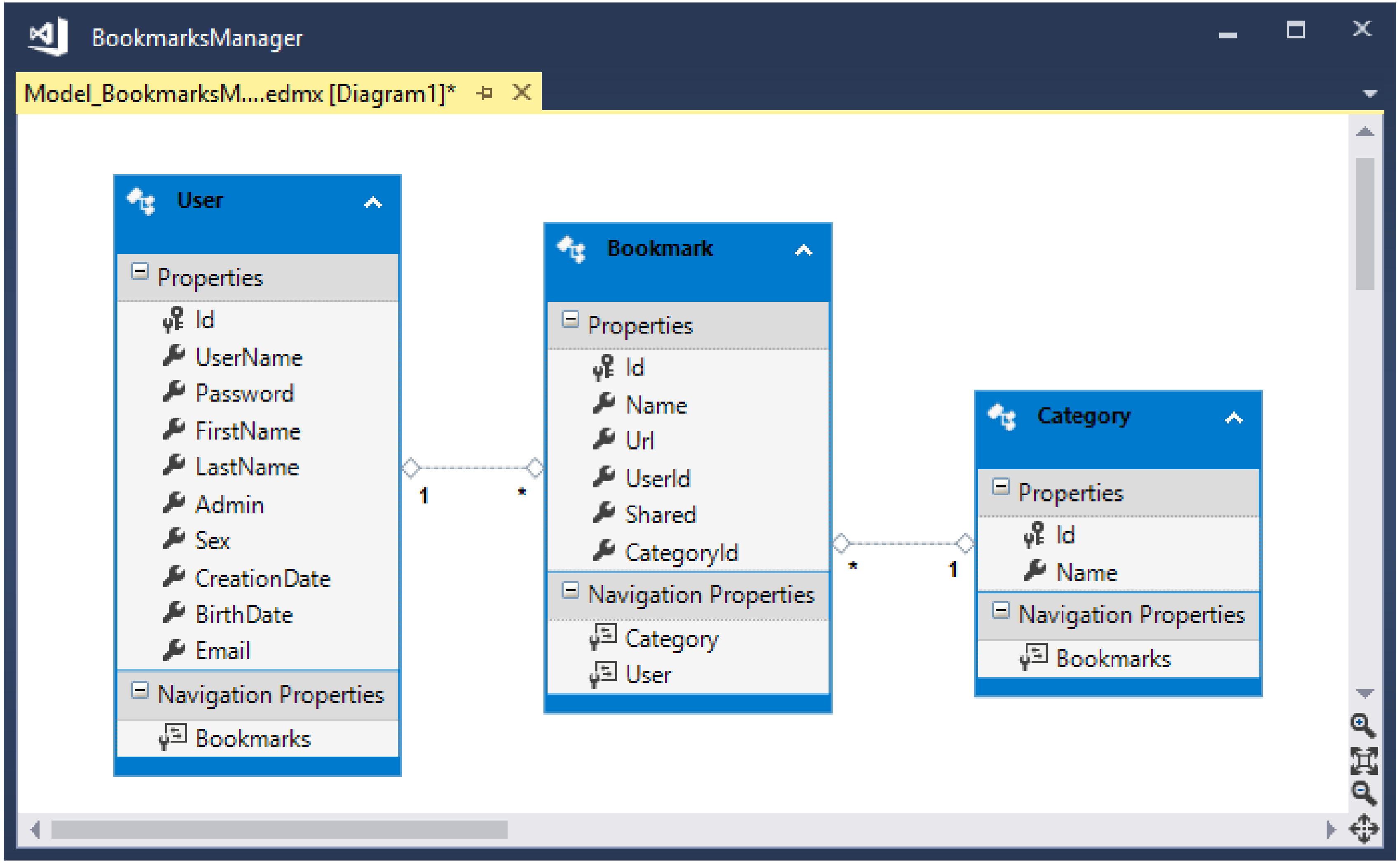
Usager connecté :

- Liste triée/filtrée des favoris
- Ajout de favoris
- Modification / retrait de favoris propriétaires
- Déconnection

Admin connecté :

- Liste triée/filtrée des favoris
- Ajout de favoris
- Modification / retrait de tous les favoris
- Voir la liste des usagers en ligne
- Retrait d'usager et de leurs favoris
- Gestion CRUD des catégories
- Déconnection

Base de données (schéma)



User
- Properties
  - Id
  - UserName
  - Password
  - FirstName
  - LastName
  - Admin
  - Sex
  - CreationDate
  - BirthDate
  - Email
- Navigation Properties
  - Bookmarks

Bookmark
- Properties
  - Id
  - Name
  - Url
  - UserId
  - Shared
  - CategoryId
- Navigation Properties
  - Category
  - User

Category
- Properties
  - Id
  - Name
- Navigation Properties
  - Bookmarks

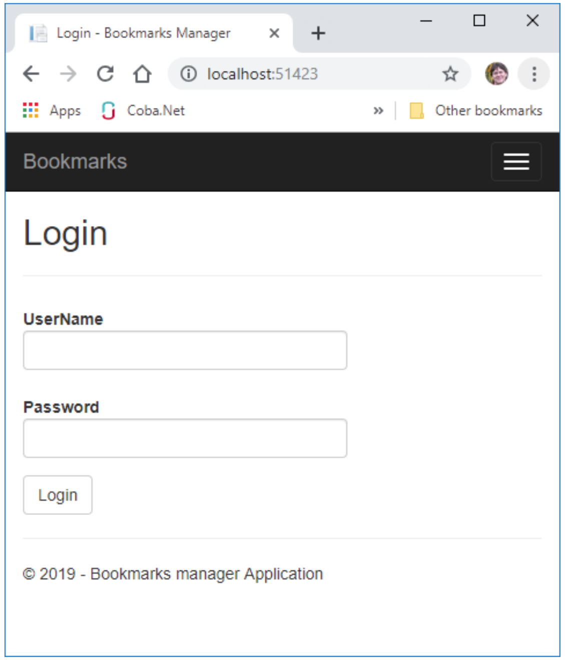
Base de données (Définitions des tables)

```
CREATE TABLE [dbo].[Users] (
    [Id]           INT            IDENTITY (1, 1) NOT NULL,
    [UserName]     NVARCHAR (MAX) NOT NULL,
    [Password]     NVARCHAR (MAX) NOT NULL,
    [FirstName]    NVARCHAR (MAX) NOT NULL,
    [LastName]     NVARCHAR (MAX) NOT NULL,
    [Admin]        BIT            NOT NULL,
    [Sex]          INT            NOT NULL,
    [CreationDate] DATETIME       NOT NULL,
    [BirthDate]    DATETIME       NOT NULL,
    [Email]        NVARCHAR (MAX) NOT NULL,
    CONSTRAINT [PK_dbo.Users] PRIMARY KEY CLUSTERED ([Id] ASC)
);

CREATE TABLE [dbo].[Bookmarks] (
    [Id]         INT            IDENTITY (1, 1) NOT NULL,
    [Name]       NVARCHAR (MAX) NOT NULL,
    [Url]        NVARCHAR (MAX) NOT NULL,
    [UserId]     INT            NOT NULL,
    [Shared]     BIT            NOT NULL,
    [CategoryId] INT            NULL,
    FOREIGN KEY ([UserId]) REFERENCES [dbo].[Users] ([Id]),
    FOREIGN KEY ([CategoryId]) REFERENCES [dbo].[Categories] ([Id]),
    CONSTRAINT [PK_dbo.Bookmarks] PRIMARY KEY CLUSTERED ([Id] ASC)
);

CREATE TABLE [dbo].[Categories] (
    [Id]   INT            IDENTITY (1, 1) NOT NULL,
    [Name] NVARCHAR (MAX) NOT NULL,
    CONSTRAINT [PK_dbo.Categories] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Users/Login

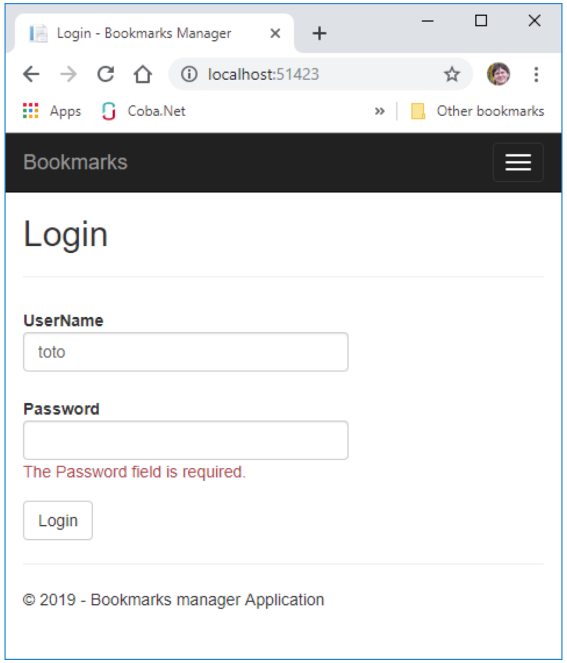


Login

UserName

Password

© 2019 - Bookmarks manager Application



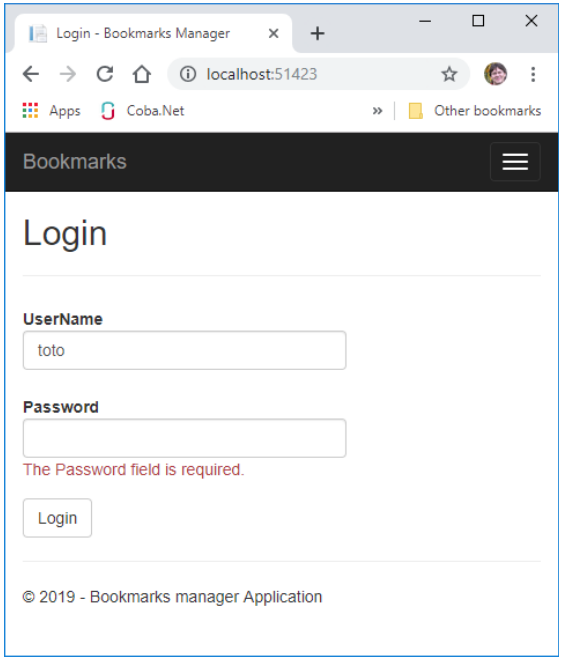
Login

UserName: toto

Password

The Password field is required.

© 2019 - Bookmarks manager Application



Login

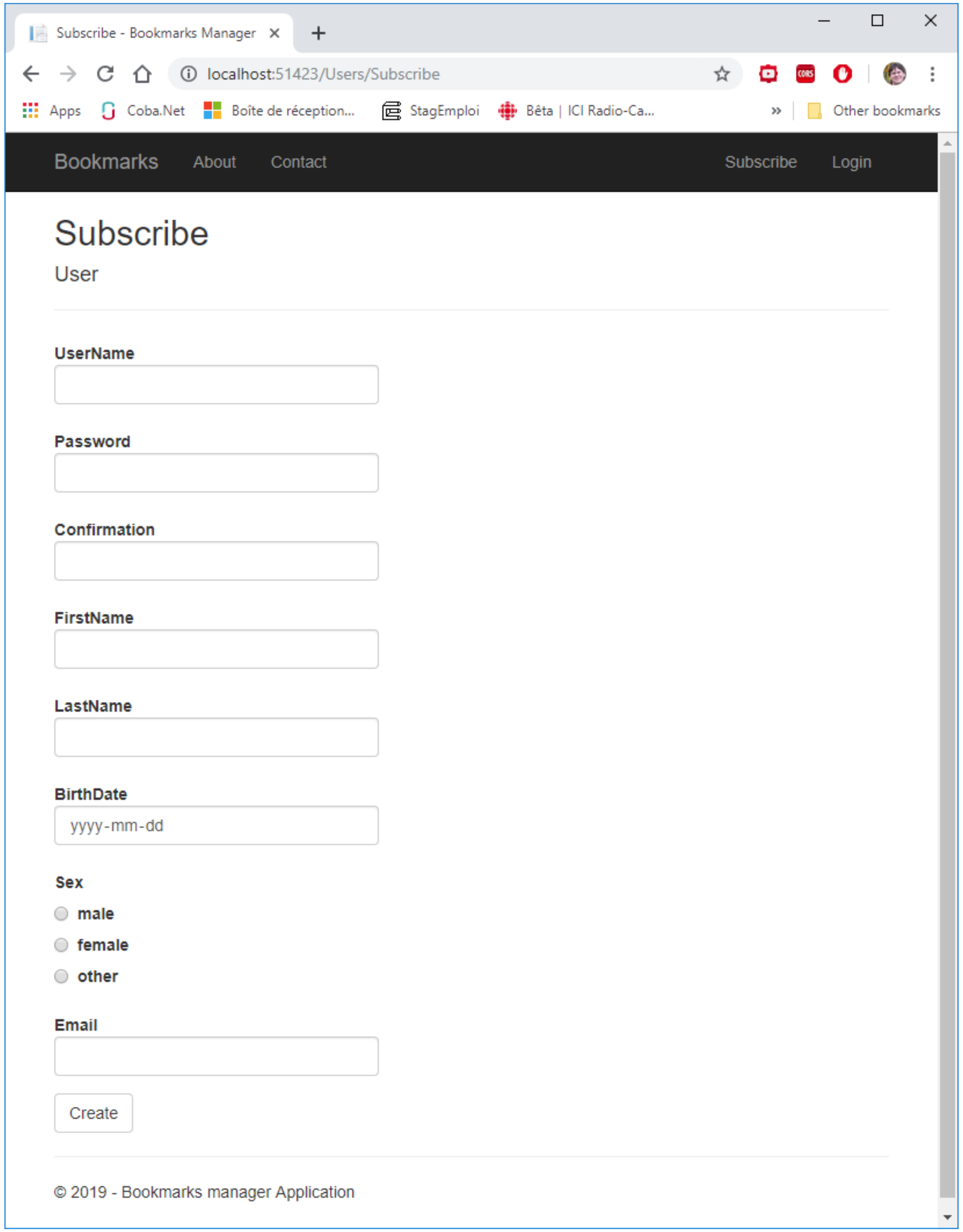
UserName: toto

Password

The Password field is required.

© 2019 - Bookmarks manager Application

Users/Subscribe



Subscribe

User

UserName

Password

Confirmation

FirstName

LastName

BirthDate

yyyy-mm-dd

Sex

○ male

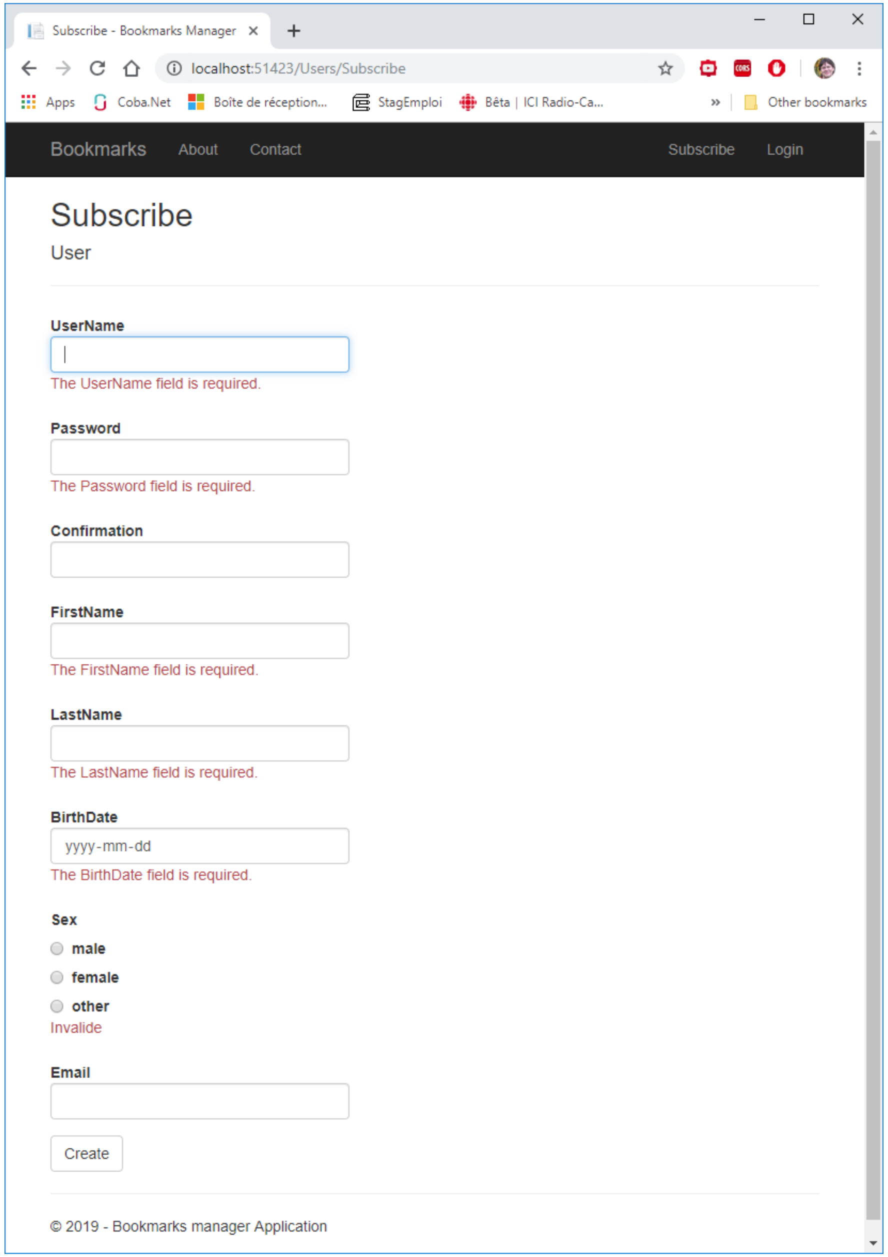
○ female

○ other

Email

Create

© 2019 - Bookmarks manager Application



Subscribe

User

UserName

The UserName field is required.

Password

The Password field is required.

Confirmation

FirstName

The FirstName field is required.

LastName

The LastName field is required.

BirthDate

yyyy-mm-dd

The BirthDate field is required.

Sex

○ male

○ female

○ other

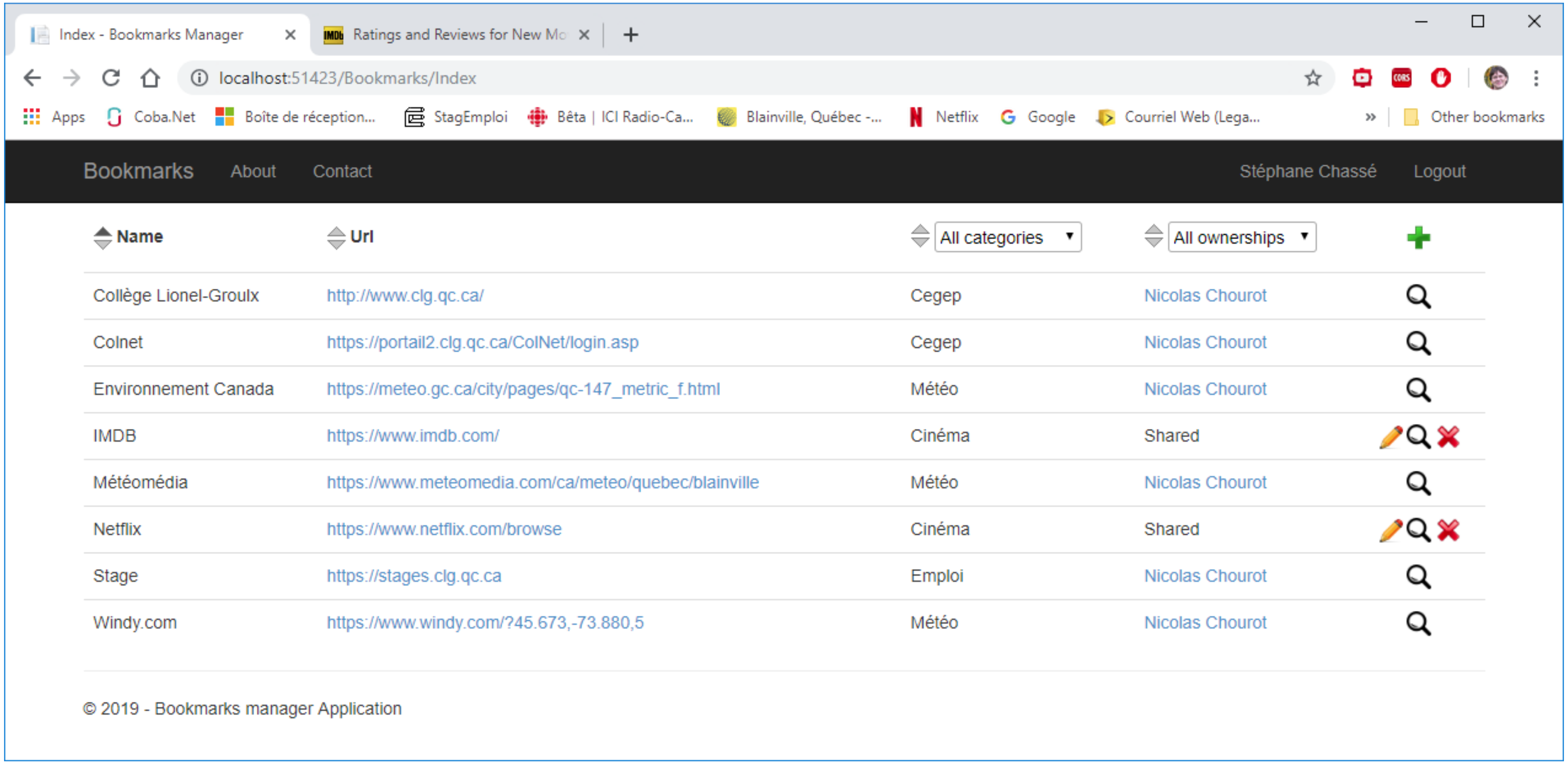
Invalide

Email

Create

© 2019 - Bookmarks manager Application

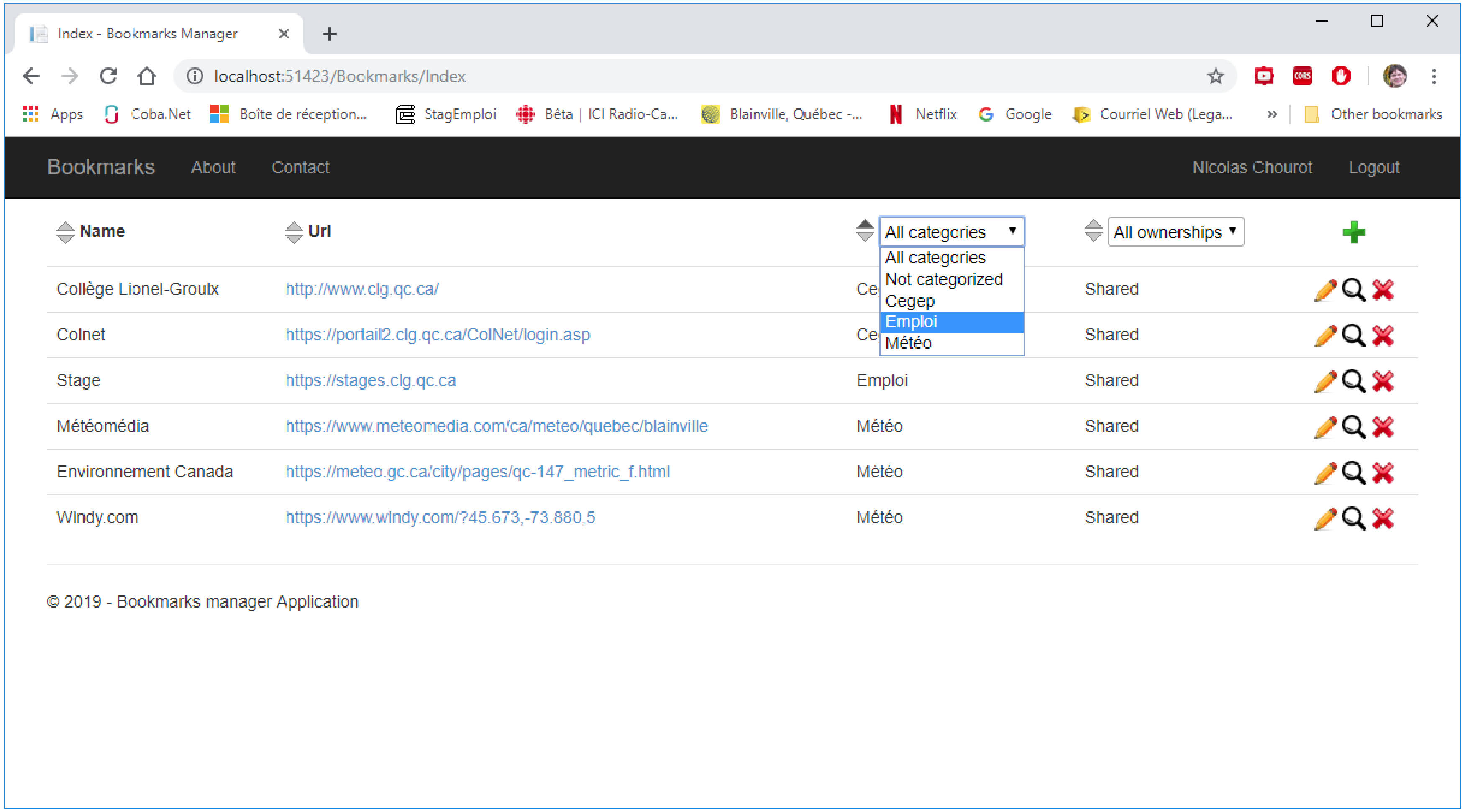
Bookmarks/Index



Name	Url	All categories ▾	All ownerships ▾	✚
Collège Lionel-Groulx	http://www.clg.qc.ca/	Cegep	Nicolas Chourot	🔍
Colnet	https://portail2.clg.qc.ca/ColNet/login.asp	Cegep	Nicolas Chourot	🔍
Environnement Canada	https://meteo.gc.ca/city/pages/qc-147_metric_f.html	Météo	Nicolas Chourot	🔍
IMDB	https://www.imdb.com/	Cinéma	Shared	✎ 🔍 ✖
Météomédia	https://www.meteomedia.com/ca/meteo/quebec/blainville	Météo	Nicolas Chourot	🔍
Netflix	https://www.netflix.com/browse	Cinéma	Shared	✎ 🔍 ✖
Stage	https://stages.clg.qc.ca	Emploi	Nicolas Chourot	🔍
Windy.com	https://www.windy.com/?45.673,-73.880,5	Météo	Nicolas Chourot	🔍

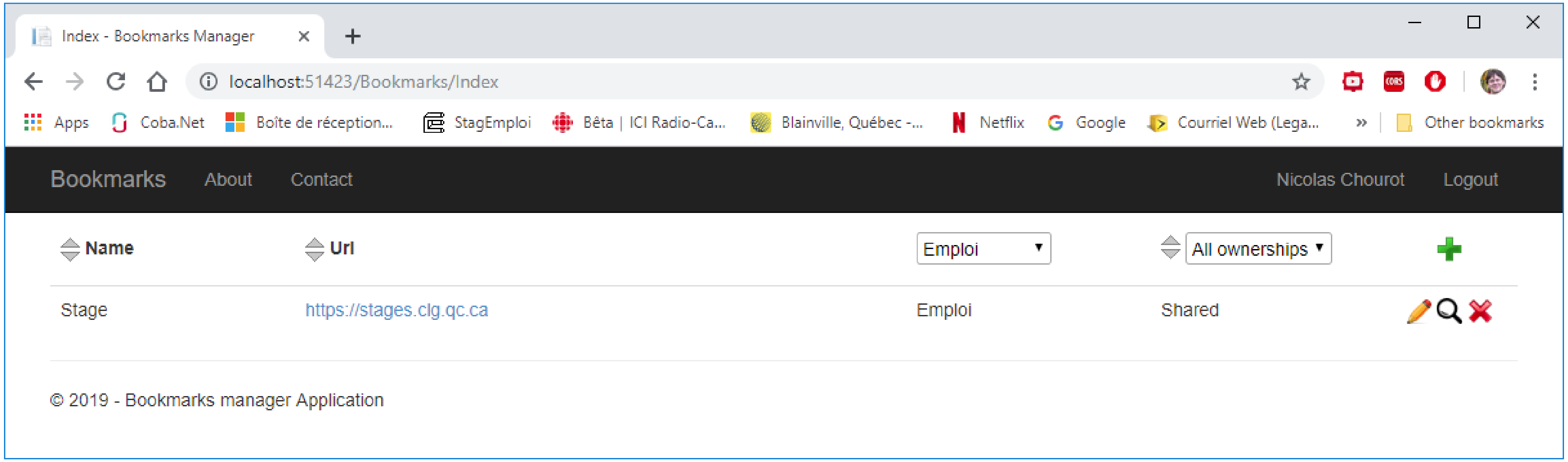
© 2019 - Bookmarks manager Application

Bookmarks/Index (filtre par catégorie)



Name	Url	All categories	All ownerships	
Collège Lionel-Groulx	http://www.clg.qc.ca/	Ce	Shared	
Colnet	https://portail2.clg.qc.ca/ColNet/login.asp	Ce	Shared	
Stage	https://stages.clg.qc.ca	Emploi	Shared	
Météomédia	https://www.meteomedia.com/ca/meteo/quebec/blainville	Météo	Shared	
Environnement Canada	https://meteo.gc.ca/city/pages/qc-147_metric_f.html	Météo	Shared	
Windy.com	https://www.windy.com/?45.673,-73.880,5	Météo	Shared	

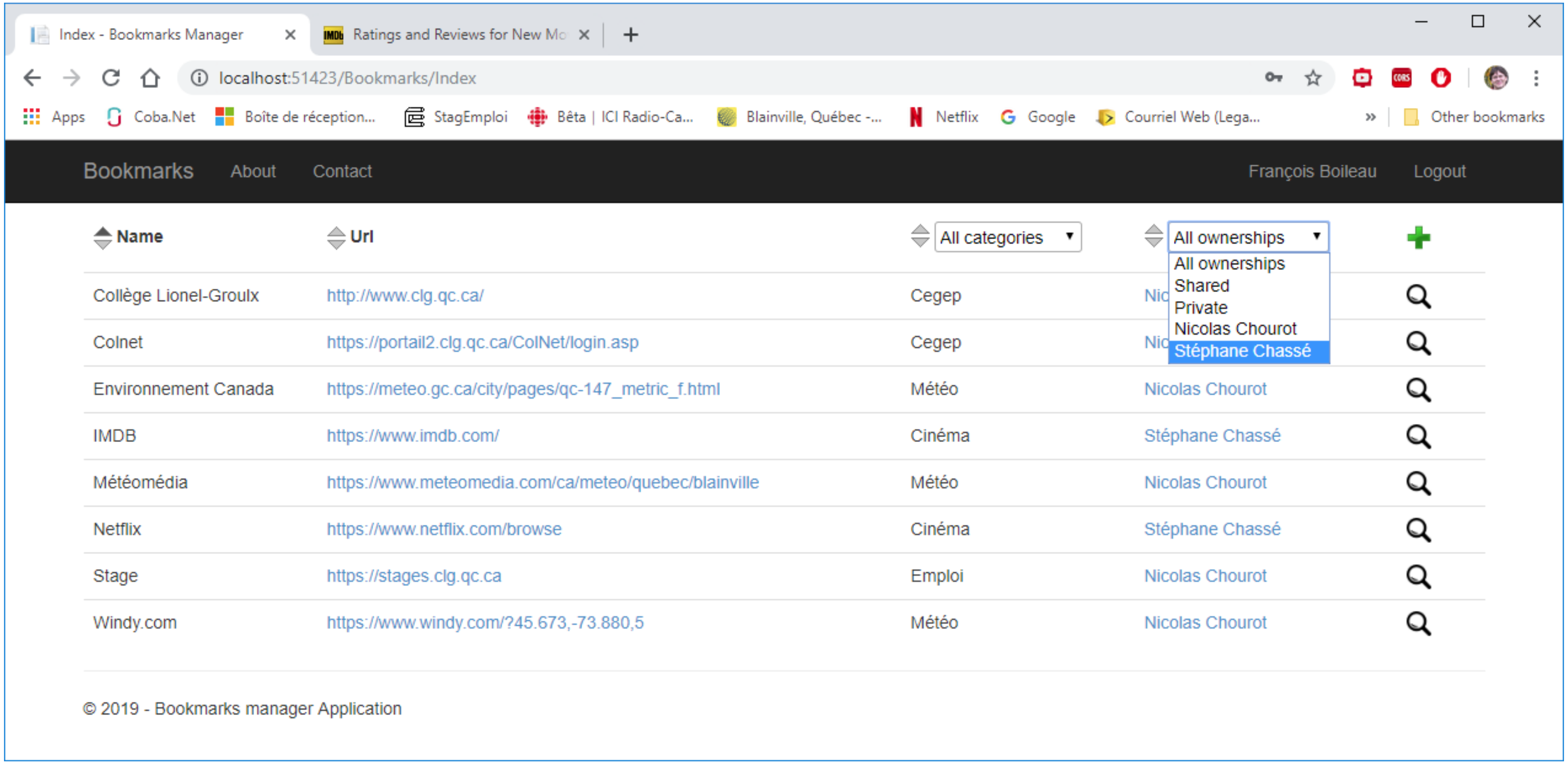
© 2019 - Bookmarks manager Application



Name	Url	Emploi ▾	All ownerships ▾	✚
Stage	https://stages.clg.qc.ca	Emploi	Shared	✎ 🔍 ✖

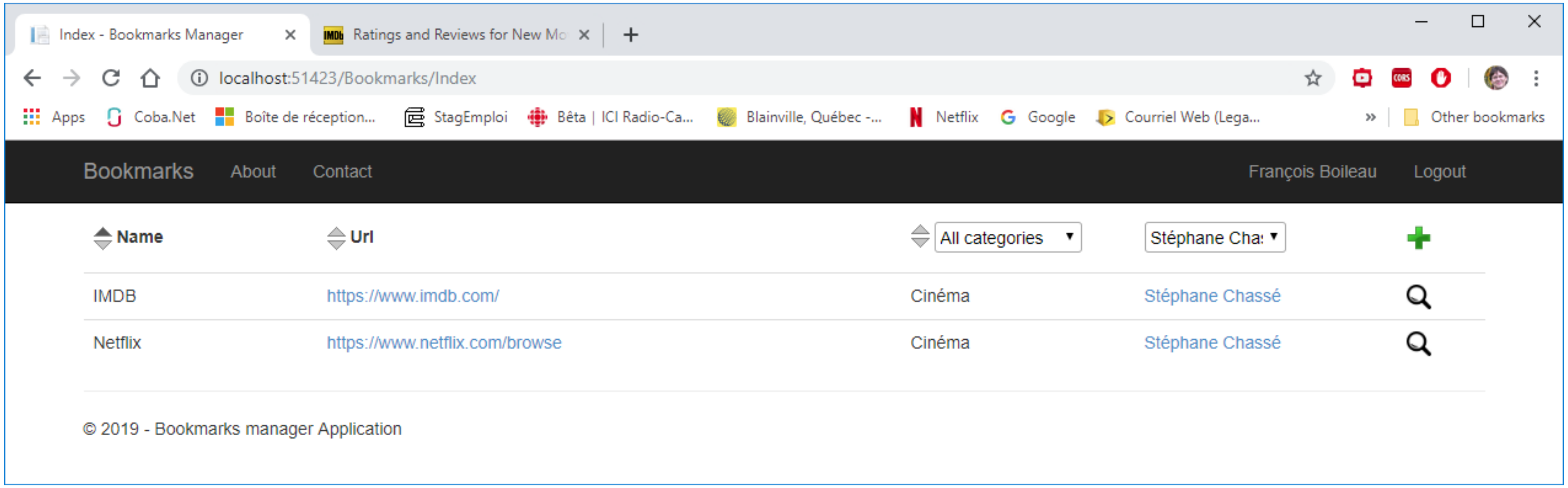
© 2019 - Bookmarks manager Application

Bookmarks/Index (filtre par propriétaires)



Name	Url	All categories	All ownerships	✚
Collège Lionel-Groulx	http://www.clg.qc.ca/	Cegep	Nic	🔍
Colnet	https://portail2.clg.qc.ca/ColNet/login.asp	Cegep	Nic	🔍
Environnement Canada	https://meteo.gc.ca/city/pages/qc-147_metric_f.html	Météo	Nicolas Chourot	🔍
IMDB	https://www.imdb.com/	Cinéma	Stéphane Chassé	🔍
Météomédia	https://www.meteomedia.com/ca/meteo/quebec/blainville	Météo	Nicolas Chourot	🔍
Netflix	https://www.netflix.com/browse	Cinéma	Stéphane Chassé	🔍
Stage	https://stages.clg.qc.ca	Emploi	Nicolas Chourot	🔍
Windy.com	https://www.windy.com/?45.673,-73.880,5	Météo	Nicolas Chourot	🔍

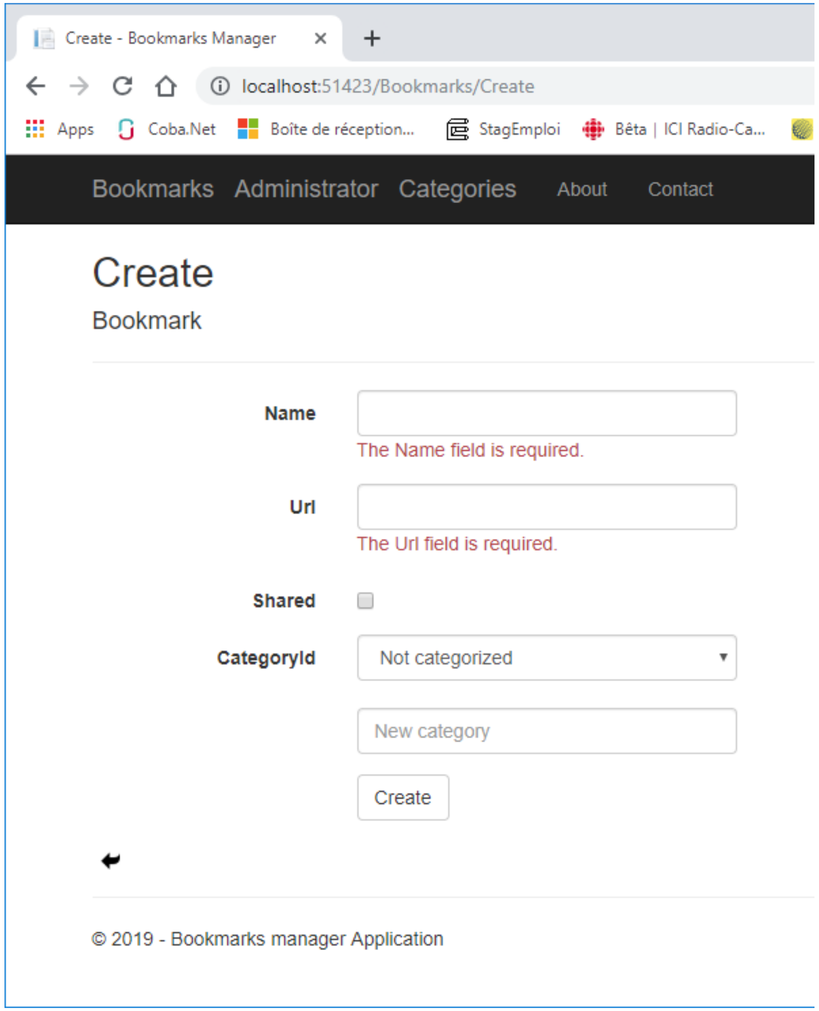
© 2019 - Bookmarks manager Application



Name	Url	All categories ▾	Stéphane Cha: ▾	✚
IMDB	https://www.imdb.com/	Cinéma	Stéphane Chassé	🔍
Netflix	https://www.netflix.com/browse	Cinéma	Stéphane Chassé	🔍

© 2019 - Bookmarks manager Application

Bookmarks/Create



Create

Bookmark

Name:

The Name field is required.

Url:

The Url field is required.

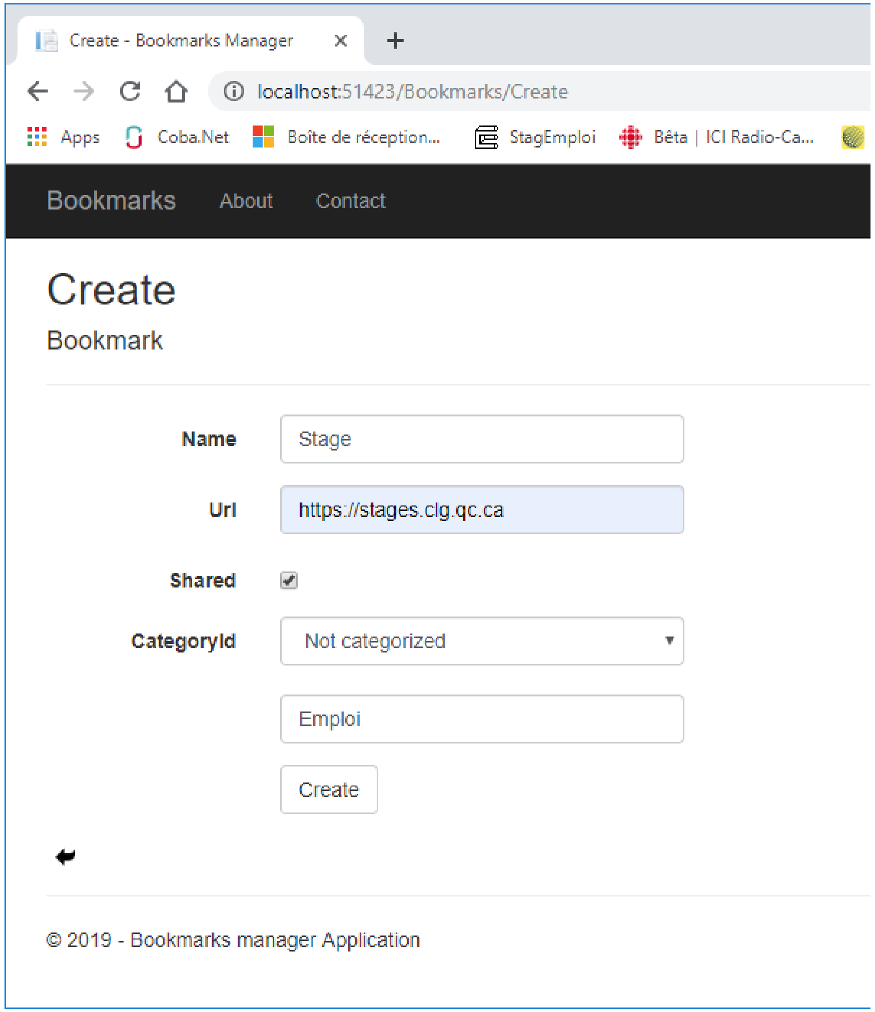
Shared: □

CategoryId: Not categorized

New category

Create

© 2019 - Bookmarks manager Application



Create

Bookmark

Name: Stage

Url: https://stages.clg.qc.ca

Shared: ☑

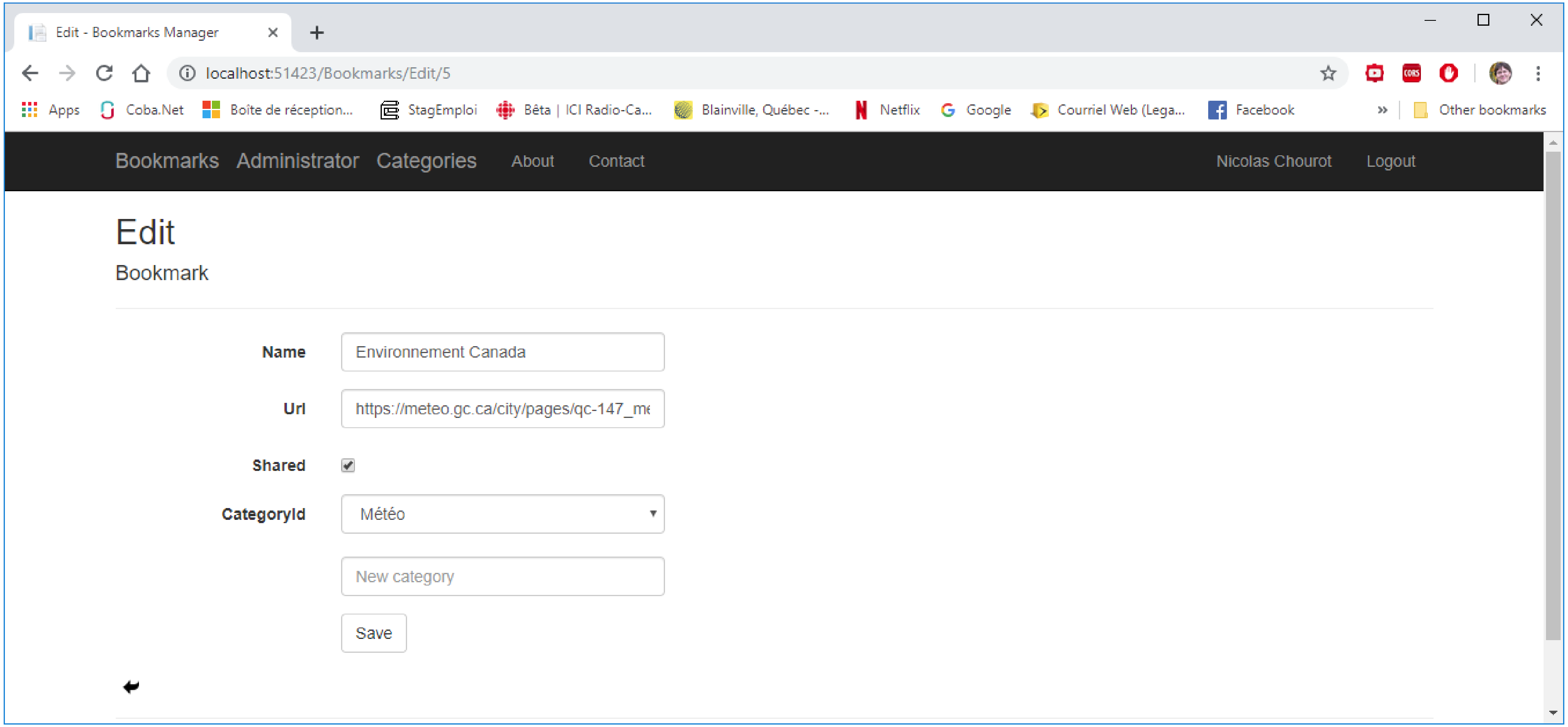
CategoryId: Not categorized

Emploi

Create

© 2019 - Bookmarks manager Application

Bookmarks/Edit

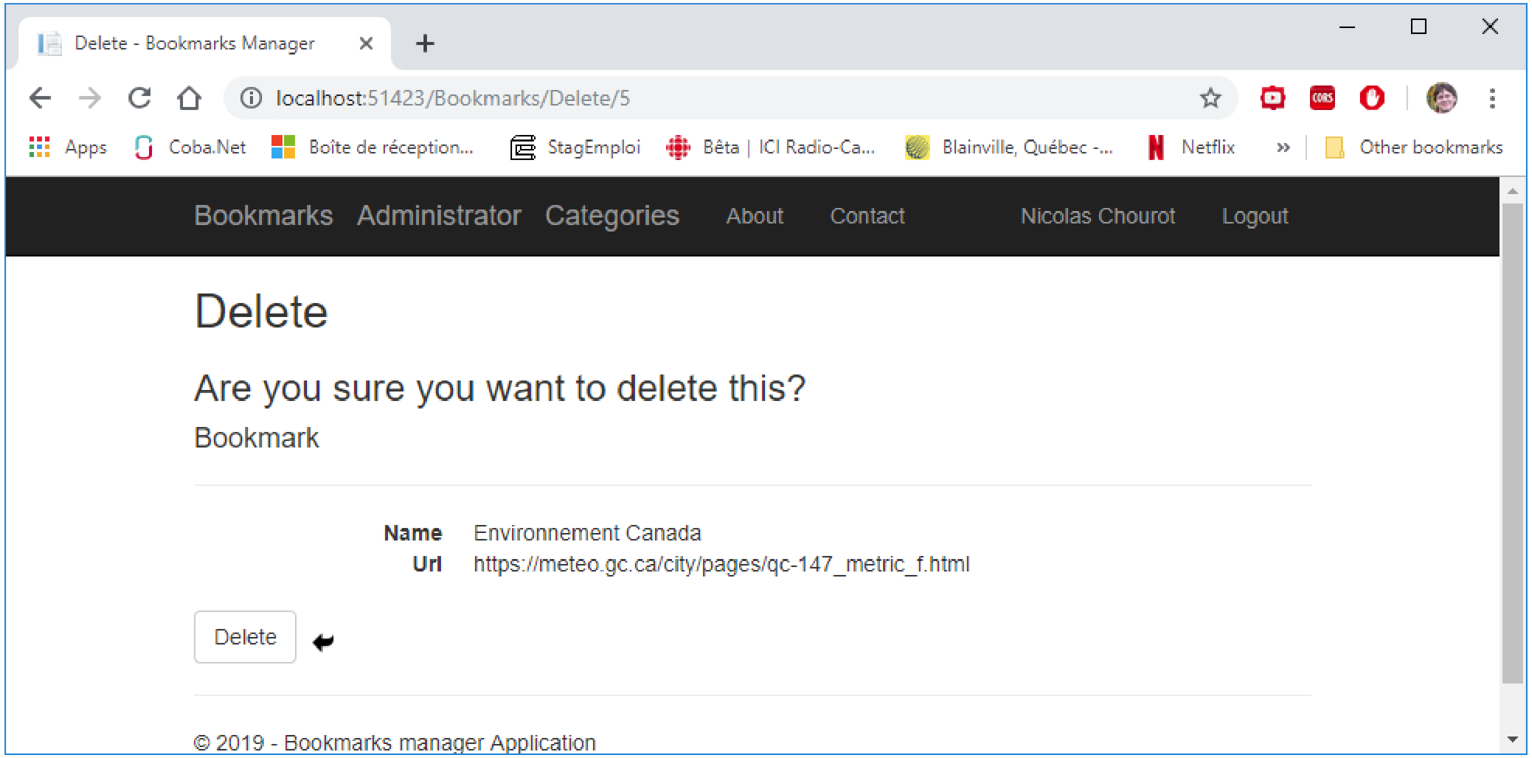


Edit

Bookmark

Name	Environnement Canada
Url	https://meteo.gc.ca/city/pages/qc-147_me
Shared	☑
CategoryId	Météo
	New category

Bookmarks/Delete/id (bookmarkId)



Delete

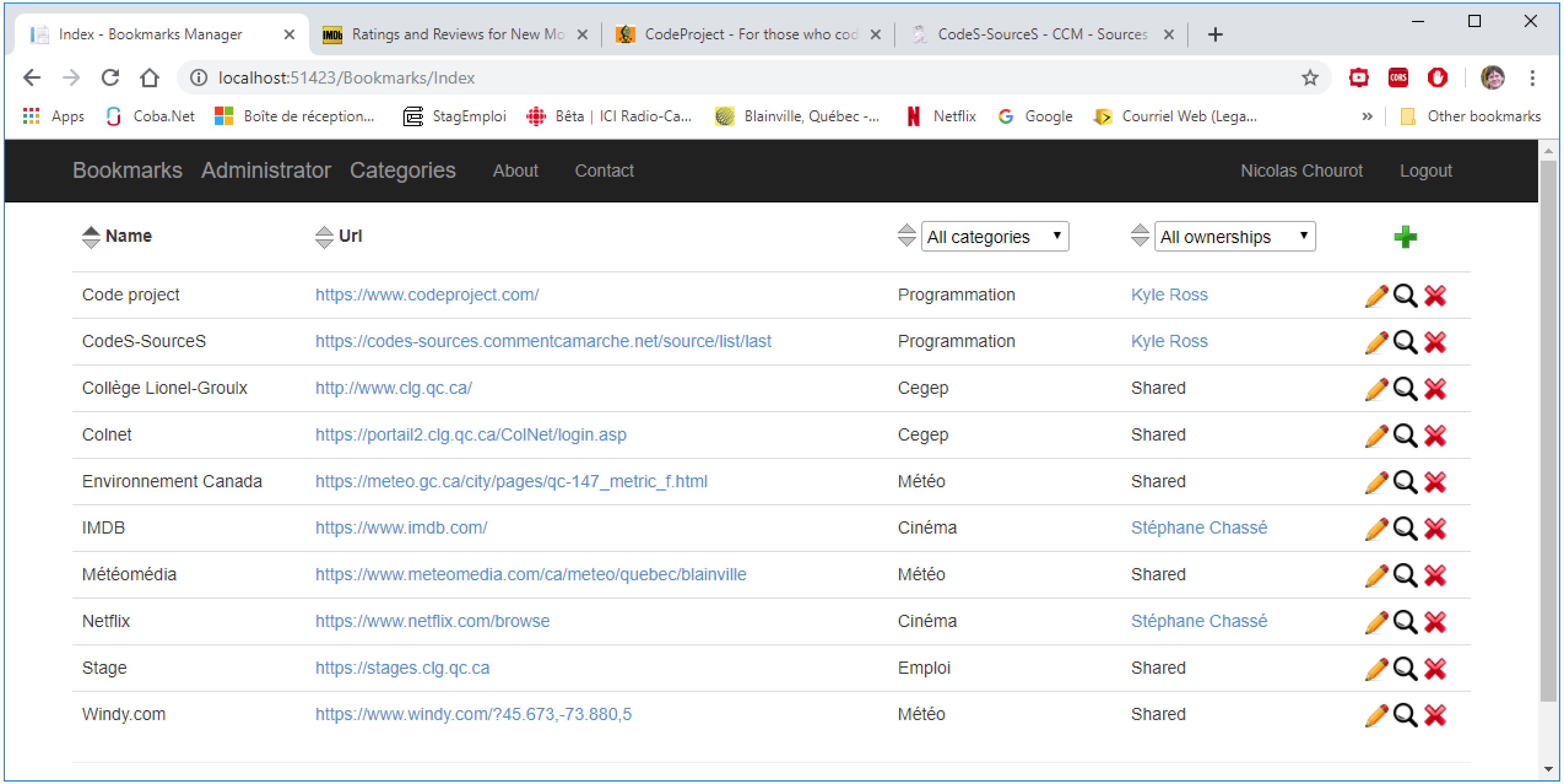
Are you sure you want to delete this?

Bookmark

Name	Environnement Canada
Url	https://meteo.gc.ca/city/pages/qc-147_metric_f.html

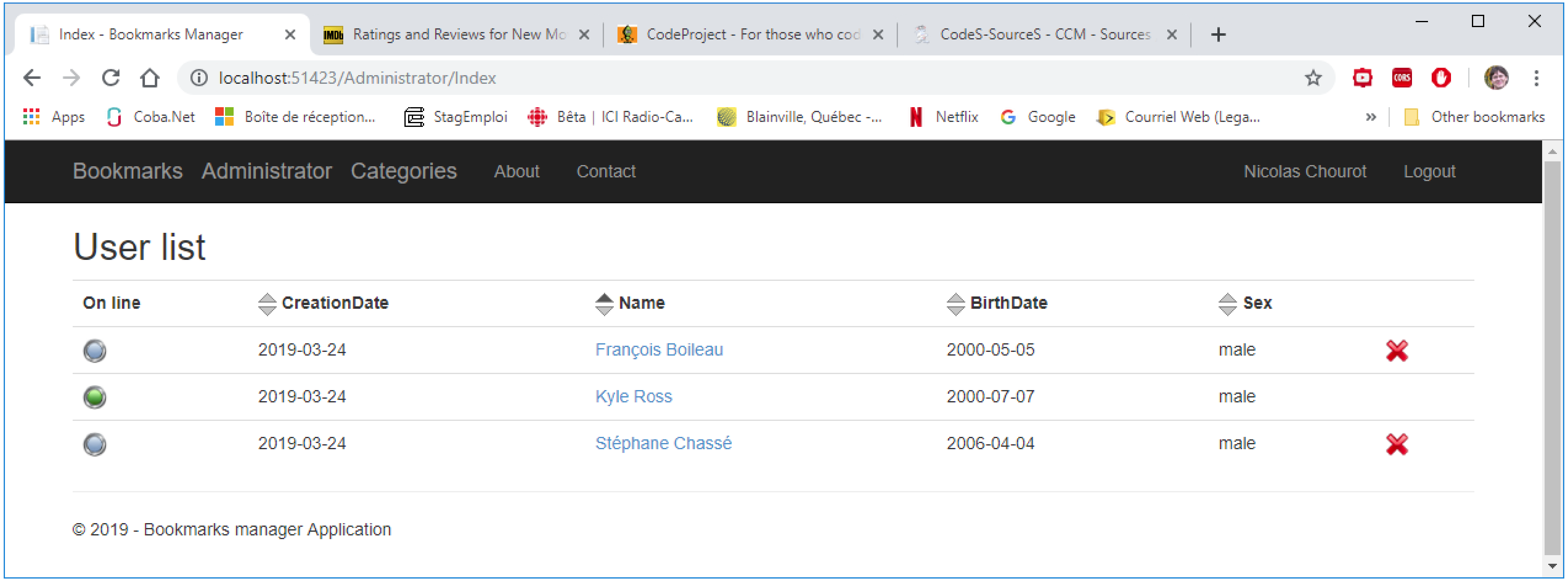
© 2019 - Bookmarks manager Application

Bookmarks/Index (rôle « admin »)



Name	Url	All categories ▾	All ownerships ▾	✚
Code project	https://www.codeproject.com/	Programmation	Kyle Ross	✎ 🔍 ✖
CodeS-SourceS	https://codes-sources.commentcamarche.net/source/list/last	Programmation	Kyle Ross	✎ 🔍 ✖
Collège Lionel-Groulx	http://www.clg.qc.ca/	Cegep	Shared	✎ 🔍 ✖
Colnet	https://portail2.clg.qc.ca/ColNet/login.asp	Cegep	Shared	✎ 🔍 ✖
Environnement Canada	https://meteo.gc.ca/city/pages/qc-147_metric_f.html	Météo	Shared	✎ 🔍 ✖
IMDB	https://www.imdb.com/	Cinéma	Stéphane Chassé	✎ 🔍 ✖
Météomédia	https://www.meteomedia.com/ca/meteo/quebec/blainville	Météo	Shared	✎ 🔍 ✖
Netflix	https://www.netflix.com/browse	Cinéma	Stéphane Chassé	✎ 🔍 ✖
Stage	https://stages.clg.qc.ca	Emploi	Shared	✎ 🔍 ✖
Windy.com	https://www.windy.com/?45.673,-73.880,5	Météo	Shared	✎ 🔍 ✖

Administrator/Index (rôle « admin »)

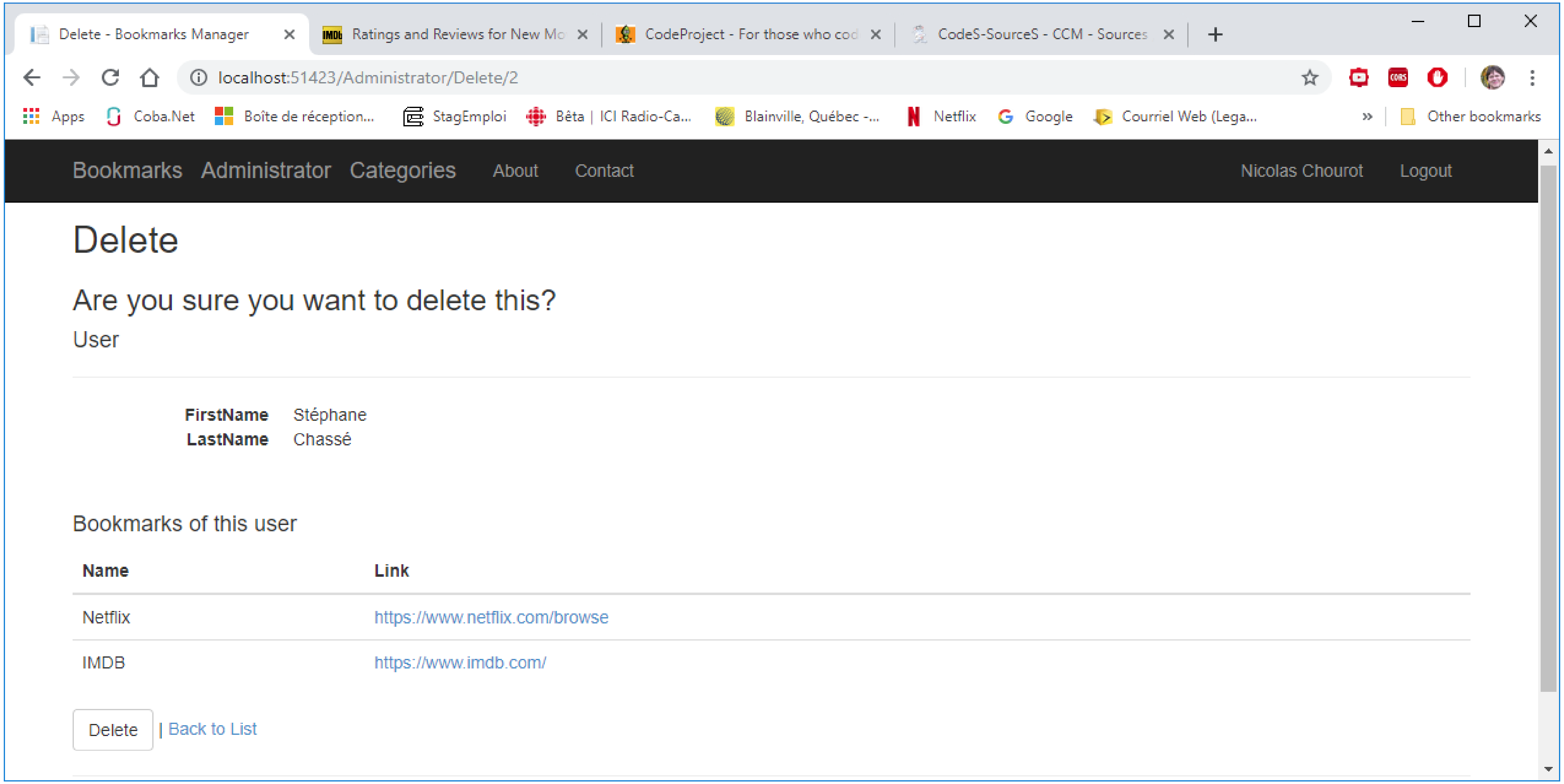


User list

On line	CreationDate	Name	BirthDate	Sex	
○	2019-03-24	François Boileau	2000-05-05	male	❌
●	2019-03-24	Kyle Ross	2000-07-07	male	
○	2019-03-24	Stéphane Chassé	2006-04-04	male	❌

© 2019 - Bookmarks manager Application

Administrator/Delete/id (userId) (rôle « admin »)



Delete

Are you sure you want to delete this?

User

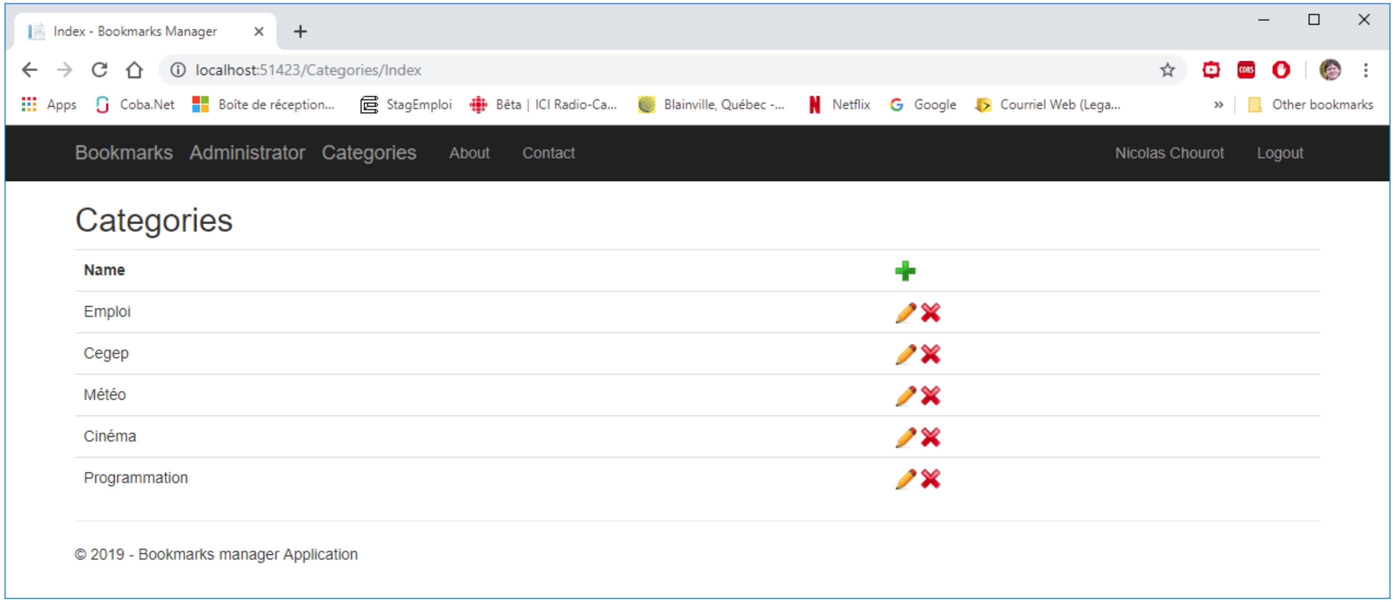
FirstName	Stéphane
LastName	Chassé

Bookmarks of this user

Name	Link
Netflix	https://www.netflix.com/browse
IMDB	https://www.imdb.com/

Delete | Back to List

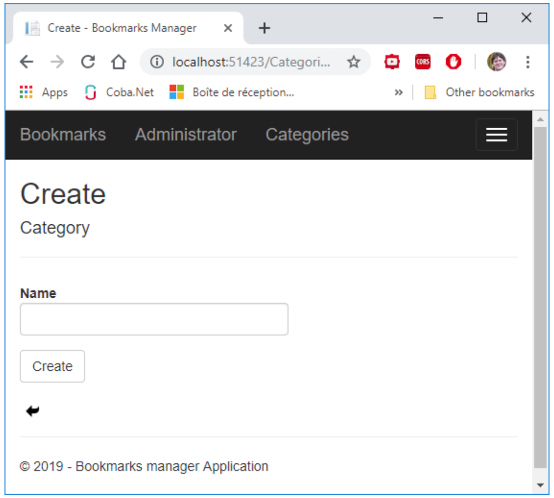
Categories/index (rôle « admin »)



Categories

Name	✚
Emploi	✎ ✖
Cegep	✎ ✖
Météo	✎ ✖
Cinéma	✎ ✖
Programmation	✎ ✖

© 2019 - Bookmarks manager Application



Create

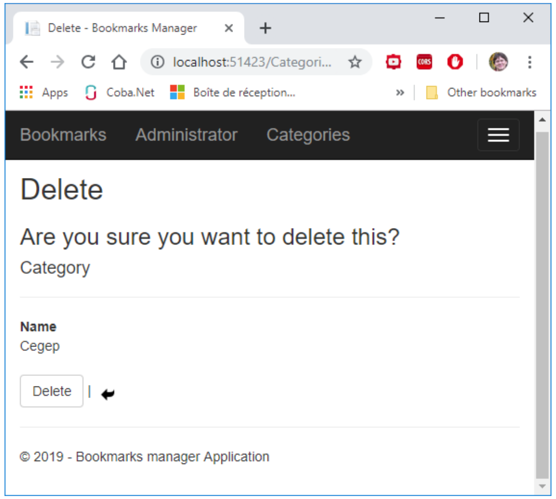
Category

Name

Create

↩

© 2019 - Bookmarks manager Application



Delete

Are you sure you want to delete this?

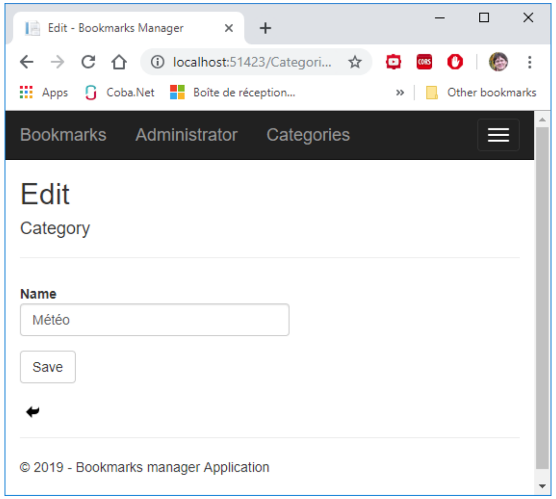
Category

Name

Cegep

Delete | ↩

© 2019 - Bookmarks manager Application



Edit

Category

Name

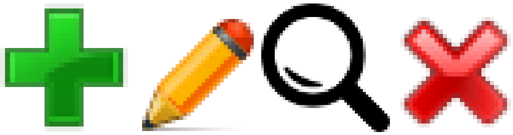
Météo

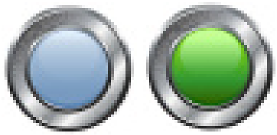
Save

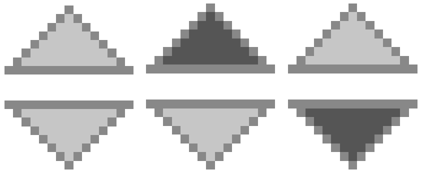
↩

© 2019 - Bookmarks manager Application

Images utiles pour l’interface (fournies)







Annexe 1 - Actions du contrôleur *Bookmarks* exploitant de Session pour mémoriser les paramètres de tri et filtres des bookmarks

(Code à inclure dans BookmarksController)

```
#region Sort and filters
private void InitSessionSortAndFilter()
{
    if (Session["BookmarkSortBy"] == null)
    {
        Session["BookmarkSortBy"] = "Name";
        Session["BookmarkSortAscendant"] = true;
    }

    if (Session["BookmarkFilterByOwnership"] == null)
    {
        Session["BookmarkFilterByOwnership"] = "";
    }

    if (Session["BookmarkFilterByCategory"] == null)
    {
        Session["BookmarkFilterByCategory"] = "All";
    }
}

public ActionResult Sort(string by)
{
    if (by == (string)Session["BookmarkSortBy"])
        Session["BookmarkSortAscendant"] = !(bool)Session["BookmarkSortAscendant"];
    else
        Session["BookmarkSortAscendant"] = true;

    Session["BookmarkSortBy"] = by;
    return RedirectToAction("Index");
}

public ActionResult FilterOwnership(string Ownership)
{
    Session["BookmarkFilterByOwnership"] = (Ownership == "All" ? "" : Ownership);
    return RedirectToAction("Index");
}

public ActionResult FilterCategory(string Category)
{
    Session["BookmarkFilterByCategory"] = Category;
    return RedirectToAction("Index");
}
#endregion
```

Annexe 2 - Extensions suggérées pour les opérations CRUD sur les tables

```
(Code à inclure dans la définition de la classe suivante)
    public class BookmarksDBEntities : DbContext
    {
        ...
        public DbSet<User> Users { get; set; }
        public DbSet<Bookmark> Bookmarks { get; set; }
        public DbSet<Category> Categories { get; set; }

    }


    //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    // Extensions for CRUD queries
    public static class BookmarksDBEntities_DAL
    {
```

```
///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Users CRUD queries

public static User Add(this BookmarksDBEntities DB, User user)
{
    if (user != null)
    {
        User new_User = DB.Users.Add(user);
        DB.SaveChanges();
        return new_User;
    }
    return null;
}

public static User Update(this BookmarksDBEntities DB, User user)
{
    if (user != null)
    {
        User userToUpdate = DB.Users.Find(user.Id);
        if (userToUpdate != null)
        {
            userToUpdate.Update(user);
            DB.Entry(userToUpdate).State = System.Data.Entity.EntityState.Modified;
            DB.SaveChanges();
            return DB.Users.Find(userToUpdate.Id);
        }
    }
    return null;
}

public static void Delete(this BookmarksDBEntities DB, User user)
{
    if (user != null)
    {
        User userToDelete = DB.Users.Find(user.Id);
        if (userToDelete != null)
        {
            foreach (Bookmark bookmark in DB.Bookmarks.Where(b => b.UserId == user.Id))
            {
                DB.Entry(bookmark).State = System.Data.Entity.EntityState.Deleted;
            }
            DB.Entry(userToDelete).State = System.Data.Entity.EntityState.Deleted;
            DB.SaveChanges();
        }
    }
}
```

```
public static User FindByUserName(this BookmarksDBEntities DB, string userName)
{
    return DB.Users.Where(u => u.UserName == userName).FirstOrDefault();
}

public static List<User> SortedUsers(this BookmarksDBEntities DB, string sortBy, bool ascendant)
{
    List<User> users = DB.Users.ToList();
    if (users.Count > 1)
    {
        switch (sortBy)
        {
            case "Name":
                if (ascendant)
                    users = users.OrderBy(b => b.FirstName).ThenBy(b => b.LastName).ToList();
                else
                    users = users.OrderByDescending(b => b.FirstName).ThenByDescending(b => b.LastName).ToList();
                break;
            case "CreationDate":
                if (ascendant)
                    users = users.OrderBy(b => b.CreationDate).ToList();
                else
                    users = users.OrderByDescending(b => b.CreationDate).ToList();
                break;
            case "BirthDate":
                if (ascendant)
                    users = users.OrderBy(b => b.BirthDate).ToList();
                else
                    users = users.OrderByDescending(b => b.BirthDate).ToList();
                break;
            case "Sex":
                if (ascendant)
                    users = users.OrderBy(b => b.Sex).ToList();
                else
                    users = users.OrderByDescending(b => b.Sex).ToList();
                break;
        }
    }
    return users;
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Bookmarks CRUD queries

public static Bookmark Add(this BookmarksDBEntities DB, Bookmark bookmark)
{
    if (bookmark != null)
    {
        Bookmark new_Bookmark = DB.Bookmarks.Add(bookmark);
        DB.SaveChanges();
        return new_Bookmark;
    }
    return null;
}

public static Bookmark Update(this BookmarksDBEntities DB, Bookmark bookmark)
{
    if (bookmark != null)
    {
        Bookmark bookmarkToUpdate = DB.Bookmarks.Find(bookmark.Id);
        if (bookmarkToUpdate != null)
        {
            bookmarkToUpdate.Update(bookmark);
            DB.Entry(bookmarkToUpdate).State = System.Data.Entity.EntityState.Modified;
            DB.SaveChanges();
            return DB.Bookmarks.Find(bookmarkToUpdate.Id);
        }
    }
    return null;
}

public static void Delete(this BookmarksDBEntities DB, Bookmark bookmark)
{
    if (bookmark != null)
    {
        Bookmark bookmarkToDelete = DB.Bookmarks.Find(bookmark.Id);
        if (bookmarkToDelete != null)
        {
            DB.Entry(bookmark).State = System.Data.Entity.EntityState.Deleted;
            DB.SaveChanges();
        }
    }
}
```

```
public static List<BookmarkItemView> BookmarkList(this BookmarksDBEntities DB, User viewer, string SortBy, bool ascendant)
{
    List<BookmarkItemView> bookmarkItems = new List<BookmarkItemView>();
    foreach (Bookmark bookmark in DB.Bookmarks)
    {
        if ((viewer.Admin) || (viewer.Id == bookmark.UserId) || (bookmark.Shared))
        {
            bookmarkItems.Add(new BookmarkItemView(DB, bookmark, viewer));
        }
    }
    switch (SortBy)
    {
        case "Name":
            if (ascendant)
                bookmarkItems = bookmarkItems.OrderBy(b => b.Name).ToList();
            else
                bookmarkItems = bookmarkItems.OrderByDescending(b => b.Name).ToList();
            break;
        case "Url":
            if (ascendant)
                bookmarkItems = bookmarkItems.OrderBy(b => b.Url).ToList();
            else
                bookmarkItems = bookmarkItems.OrderByDescending(b => b.Url).ToList();
            break;
        case "OwnerShip":
            if (ascendant)
                bookmarkItems = bookmarkItems.OrderBy(b => b.OwnerShip).ToList();
            else
                bookmarkItems = bookmarkItems.OrderByDescending(b => b.OwnerShip).ToList();
            break;
        case "Category":
            if (ascendant)
                bookmarkItems = bookmarkItems.OrderBy(b => b.Category).ToList();
            else
                bookmarkItems = bookmarkItems.OrderByDescending(b => b.Category).ToList();
            break;
    }

    return bookmarkItems;
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Categories CRUD queries

public static Category Add(this BookmarksDBEntities DB, Category category)
{
    if (category != null)
    {
        Category new_Category = DB.Categories.Add(category);
        DB.SaveChanges();
        return new_Category;
    }
    return null;
}
public static Category Update(this BookmarksDBEntities DB, Category category)
{
    if (category != null)
    {
        Category categoryToUpdate = DB.Categories.Find(category.Id);
        if (categoryToUpdate != null)
        {
            categoryToUpdate.Update(category);
            DB.Entry(categoryToUpdate).State = System.Data.Entity.EntityState.Modified;
            DB.SaveChanges();
            return DB.Categories.Find(categoryToUpdate.Id);
        }
    }
    return null;
}
public static void Delete(this BookmarksDBEntities DB, Category category)
{
    if (category != null)
    {
        Category categoryToDelete = DB.Categories.Find(category.Id);
        if (categoryToDelete != null)
        {
            foreach (Bookmark bookmark in DB.Bookmarks.Where(b => b.CategoryId == categoryToDelete.Id))
            {
                bookmark.CategoryId = 0;
                DB.Entry(bookmark).State = System.Data.Entity.EntityState.Modified;
            }
            DB.Entry(categoryToDelete).State = System.Data.Entity.EntityState.Deleted;
            DB.SaveChanges();
        }
    }
}
```

```
    public static bool CategoryExist(this BookmarksDBEntities DB, string name)
    {
        foreach (Category category in DB.Categories)
        {
            if (category.Name == name)
                return true;
        }
        return false;
    }
}
```

Annexe 3 - HTML Helper

RadioButtonsGroupFor (inclure le source CustomHelper.cs dans le répertoire App_Code)

```
<div class="col-md-10">
	@Html.RadioButtonsGroupFor(model => model.Sex, true /*creating*/)
	@Html.ValidationMessageFor(model => model.Sex, "", new { @class = "text-danger" })
</div>
```

Helper.SortAction (inclure le source Helper.cshtml dans le répertoire App_Code)

```
@Helper.SortAction(Url, "Sort", Html.DisplayNameFor(model => model.Name).ToString(), "Bookmark", "Name")
```